Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

DESENVOLVIMENTO DE UMA APLICAÇÃO *WEB* PARA CONTROLE E GERENCIAMENTO DE PERÍODOS AQUISITIVOS, FRUITIVOS E QUINQUÊNIOS

DEVELOPING WEB APPLICATION FOR CONTROLLING AND MANAGEMENT OF ACQUISITIVE, FRUITIVE AND QUINQUENNIUM PERIODS

João Marcos Polo Junior[1] Thiago Santos Mota[2]

RESUMO

O uso de planilhas eletrônicas se tornou muito difundido na vida moderna, principalmente nas áreas administrativas. Muito se pode alcançar em termos de produtividade e gerenciamento. Entretanto, quando a quantidade de dados aumenta ou quando é necessário agregar funções de automatização e cálculos complexos, essas planilhas não conseguem satisfazer às necessidades dos usuários de maneira simples ou os próprios usuários não possuem o conhecimento necessário para desenvolver uma solução que resolva um problema. O objetivo deste estudo foi desenvolver uma solução para o controle e gerenciamento de períodos aquisitivos, fruitivos e quinquênios, abstraindo todas as regras de negócio de algumas planilhas do setor de recursos humanos da Fatec de Botucatu e incorporando todas as vantagens de um sistema, como validação e consistência de dados e envio automático de e-mails. Ao final do desenvolvimento, todas as regras e cálculos presentes nas planilhas foram traduzidos para uma aplicação *web* com banco de dados e interface gráfica, que valida e persiste os dados de forma lógica, enviando e-mails automáticos, quando regras pré-determinadas são satisfeitas.

Palavras-chave: Aplicação web. E-mails automáticos. Planilhas eletrônicas.

ABSTRACT

The use of spreadsheets has become very widespread in modern life, especially in the administrative area. Much can be achieved in terms of productivity and management. However, when the amount of data increase or when it is necessary to add automation functions and complex calculations, these spreadsheets cannot meet the needs of users in a simple way or the users themselves do not have the necessary knowledge to develop a solution that solves a problem. The goal of this study was to develop a web application for the control and management of acquisition, fruitive and five-year periods, abstracting all the business rules of some spreadsheets in the human resources area of an university and incorporating all the advantages of a system, such as validation and data consistency and automatic emailing. At the end of the development, all rules and calculations present in the spreadsheets were translated into a web application with a database and graphical interface, which validates and persists the data in a logical way, sending automatic emails when predetermined rules are satisfied.

Key Words: Automatic e-mail. Spreadsheets. Web application.

[1] Tecnólogo em Análise e Desenvolvimento de Sistemas, FATEC Botucatu. Av. José Ítalo Bacchi, s/n - Jardim Aeroporto, Botucatu - SP, 18606-851. e-mail: jpolojunior@gmail.com
[2] Professor de Ensino Superior do Centro Paula Souza, FATEC Botucatu

Recebido em novembro de 2021 e Aceito em março de 2022.

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

1 INTRODUÇÃO

Softwares de computador continuam a ser a tecnologia mais importante no cenário mundial. Há 60 anos ninguém poderia prever que o *software* se tornaria uma tecnologia indispensável para negócios, ciência e tecnologia (PRESSMAN; MAXIM, 2016).

Softwares de planilhas eletrônicas, como por exemplo o Microsoft Excel, foram desenvolvidos por pessoas que perceberam a dificuldade de outros em modificar e realizar, no quadro negro, cálculos dispostos em linhas e colunas, que precisavam ser atualizados manualmente quando a variável de uma determinada célula mudava de valor. Uma planilha pode ser utilizada por qualquer profissional que tenha no seu trabalho a necessidade de efetuar cálculos matemáticos, financeiros, estatísticos ou científicos (SANTOS, 2011).

Na área de recursos humanos existe a necessidade de se administrar quinquênios. Cada quinquênio corresponde a um período de cinco anos, pelo qual o servidor público recebe um adicional de 5% sobre seu cargo efetivo (BOTUCATU, 2011). Nesse mesmo âmbito é preciso gerenciar períodos aquisitivos e fruitivos, sendo que o colaborador, após trabalhar por 12 meses, tem até 30 dias de férias remuneradas usufruídas dentro de um ano e 11 meses (BRASIL, 1943).

Há a necessidade de muito investimento em capacitação específica, para que usuários potenciais possam se familiarizar com dispositivos (ou ferramentas) digitais. E capacitação não é um bem que pode ser adquirido de imediato, havendo a necessidade de tempo para assimilação da informação e geração do conhecimento (CARVALHO, 2003).

Apesar de todo esforço e curva de aprendizado percorrida para se confeccionar uma planilha e converter regras de negócio em fórmulas matemáticas, configurar uma planilha em seus mínimos detalhes para que ela se assemelhe a um sistema, com regras bem definidas, imutáveis, travando as células e as planilhas, criando e associando macros a eventos de botões, definindo as visualizações e principalmente mantendo a confiabilidade dos dados é algo que até mesmo usuário com mais conhecimento não conseguem atingir. Além disso, algumas funcionalidades não podem ser implementadas simplesmente pelas limitações da aplicação utilizada. Sendo assim, o objetivo deste trabalho foi desenvolver uma solução que incluísse essas regras de quinquênios, períodos aquisitivos e fruitivos em uma aplicação *web*, abolindo o uso das planilhas, oferecendo todas as funcionalidades e benefícios de um sistema e eliminando a curva de aprendizado que o usuário teria que percorrer se quisesse desenvolver uma planilha completa e com todas as características de um sistema.

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

2 MATERIAL E MÉTODOS

2.1 API'S Rest

Serviços *web* são construídos para dar suporte às necessidades de um site ou qualquer outra aplicação. Diversos sistemas usam as apis (interfaces de programação de aplicações, em inglês) para se comunicar com esses serviços e servidores. De um modo geral uma API expõe um conjunto de dados e funções que facilitam a integração entre diferentes plataformas e permite que elas troquem informações entre si. Uma api REST é uma api construída seguindo diversos padrões de desenvolvimento que definem normas e regras para a construção de uma api que pode ser aproveitada por diversos tipos de clientes (MASSE, 2011).

Para esse desenvolvimento, os principais conceitos de uma api REST foram aplicados, como o uso correto de rotas HTTP, a montagem de URLs e o desenvolvimento de respostas em formato JSON (FIGURA 1). Isso possibilita que qualquer tipo de interface gráfica no cliente, seja ele um *smartphone* ou um sistema *desktop* consuma os dados e abre portas para que a aplicação evolua sem que seu núcleo, onde todas as regras de negócio residem tenha que ser reformulado.

Figura 1 – Exposição das rotas e das funções usando os métodos HTTP corretos.

```
14  //COMPANY
15  routes.get('/company', CompanyController.index);
16
17  routes.post('/company', celebrate({
18      [Segments.BODY]: Joi.object().keys({
19          name: Joi.string().required().max(60)
20      })
21  }), CompanyController.create);
22
23  routes.put('/company/:id', celebrate({
24      [Segments.BODY]: Joi.object().keys({
25          name: Joi.string().required().max(60)
26      }),
27      [Segments.PARAMS]: {
28          id: Joi.number().required()
29      }
30  }), CompanyController.edit);
31
32  routes.delete('/company/:id', celebrate({
33      [Segments.PARAMS]: {
34          id: Joi.number().required()
35      }
36  }), CompanyController.delete);
37
38  routes.get('/company/:id', CompanyController.findById);
39
40  routes.get('/company/:id/employees', CompanyController.employees);
```

Fonte: O autor, 2020.

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

2.1 Node.Js, Express e Javascript

O Node.Js é um ambiente de execução em tempo real de Javascript, projetado para criar aplicações *web* escaláveis (NODE.JS, 2020). Sistemas para *web* desenvolvidos nativamente sobre plataformas como PHP e Java paralisam o processamento quando estão trabalhando com operações de entrada e saída como por exemplo enviar e-mails, consultar o banco de dados e manipular arquivos. Por ser orientado a eventos, a cada iteração ele verifica se um evento foi emitido e então o executa, aguardando o momento em que precisará executar o evento de resposta (*callback*) (PEREIRA, 2014).

É uma ferramenta enxuta e não pode ser comparada com outros servidores como o Apache ou Tomcat. Sua arquitetura permite a instalação de módulos que agregam mais funções ao ambiente. O Express é um *framework* que executa em cima do Node.JS como um módulo e facilita a organização das funcionalidades da aplicação, configurando o roteamento, tratando requisições e respostas e o fornecimento de conteúdos estáticos e dinâmicos (HAHN, 2016) (FIGURA 2).

Para finalizar o conjunto de tecnologias de desenvolvimento, o Javascript é a linguagem de programação da *web*. A ampla maioria dos *sites* modernos usa Javascript e todos os navegadores modernos incluem interpretadores Javascript. Essa linguagem faz parte da tríade de tecnologias que todos os desenvolvedores *web* devem conhecer, sendo o Javascript utilizado para especificar o comportamento das páginas (FLANAGAN, 2013). Essa linguagem sem dúvida evoluiu muito e agora com o Node.Js ela pode ser utilizada no lado do servidor, tornando-a indispensável para o desenvolvimento do sistema.

Figura 2 – Node.Js, Express e Javascript integrados no desenvolvimento *back-end*.

```
//Usando o modulo de rotas do express;
const express = require('express');
const routes = express.Router();

const { celebrate, Segments, Joi } = require('celebrate');

const CompanyController = require('./controllers/CompanyController');
const EmployeeController = require('./controllers/EmployeeController');
const QuinquenniumController = require('./controllers/QuinquenniumController');
const AcquisitionController = require('./controllers/AcquisitionController');
const FruitionController = require('./controllers/FruitionController');
const EmailController = require('./controllers/EmailsControllers');

//COMPANY
routes.get('/company', CompanyController.index);

routes.post('/company', celebrate({
    [Segments.BODY]: Joi.object().keys({
        name: Joi.string().required().max(60)
    })
}), CompanyController.create);

```

Fonte: O autor, 2020.

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

2.2 Sqlite3 e Knex

O SQLite é um software de domínio público que fornece um sistema de gerenciamento de banco de dados relacionais. Dentro de suas vantagens está o fato de ele não precisar de um servidor separado para funcionar e não necessitar de configurações adicionais. Toda a base de dados é integrada junto com a aplicação em um único arquivo sem a necessidade de montar uma infraestrutura para o seu funcionamento. Essa simplicidade permite incluir o SQLite em qualquer ambiente como *smartphones*, reprodutores de música e *video games*. O arquivo gerado por uma base de dados SQLite contém as definições de tabelas e os dados em si, o que permite fazer *backup's* e restaurações simplesmente copiando e colando o arquivo (KREIBICH, 2010).

No desenvolvimento desse projeto foi usado o Knex para interagir com o banco de dados. Essa é uma biblioteca desenhada para construir consultas SQL, utilizando a sintaxe do Javascript (KNEX.JS, 2020). Ela provê uma camada de abstração para diversos bancos de dados e unifica as consultas SQL sem que os desenvolvedores tenham que se preocupar com as pequenas variações de sintaxe e formato de resposta que existem entre as plataformas (UZAYR *et al.*, 2019). Todas as consultas construídas são traduzidas para o banco de dados utilizado no desenvolvimento (FIGURA 3), flexibilizando a base de dados e permitindo migrar de banco de dados sem complicação.

Figura 3 – Consultas SQL abstraídas em funções Javascript com Knex.

```
104    async acquisitions(request, response) {
105        const { id } = request.params;
106        const employee = await connection('employee')
107                                .select(['employee.*', 'company.name AS company_name'])
108                                .leftOuterJoin('company', 'employee.company_id', 'company.id')
109                                .where({ "employee.id": id })
110                                .first();
111
112        //Query traduzida para SQLite:
113        //SELECT employee.*, company.name AS company_name
114        //FROM employee LEFT OUTER JOIN company ON employee.company_id = company.id
115        //WHERE employee.id = id
116        //LIMIT 1;
117
118        //Query traduzida para SQL SERVER:
119        //SELECT TOP 1 employee.*, company.name AS company_name
120        //FROM employee LEFT OUTER JOIN company ON employee.company_id = company.id
121        //WHERE employee.id = id;
122
123        if (employee) {
124            const acquisitions = await connection('acquisition').select('*').where({ employee_id: id }).orderBy('begin', 'asc');
125            return response.json(acquisitions);
126        } else {
127            return response.status(404).send({ errors: [{ 0: 'No employee found with the id provided.' }] });
128        }
129    },
```

Fonte: O autor, 2020.

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

2.3 E-Mail Marketing, Node Mailer e Node-Cron

Quando se coloca um sistema no ar, espera-se que “o mundo” ficará sabendo sobre ele e avidamente começará a usá-lo. Entretanto, num mundo veloz e informatizado se faz necessário uma ação efetiva de marketing para buscar usuários e fidelizá-los. O uso do e-mail como canal de comunicação é prático, barato e possuí uma relação custo/benefício extremamente favorável. Uma das aplicações do uso dos e-mails em um sistema é denominada “e-mail de rotina” e se caracteriza por mensagens simples, normalmente elaboradas pelo próprio desenvolvedor, que são enviadas automaticamente quando algum gatilho no sistema é satisfeito, como por exemplo: a chegada de uma data, um valor atingido ou uma configuração alterada. Esses e-mails promovem segurança, confiabilidade e comodidade no uso do sistema (FELIPINI, 2014).

Para o envio de e-mail foi-se utilizado um módulo do Node.Js chamado Node Mailer. Esse módulo permite configurar e enviar e-mails usando servidores proprietários ou serviços de e-mail comuns como o Gmail (FIGURA 4). Dentre suas características estão a possibilidade de configurar mensagem em HTML e manusear anexos (NODEMAILER, 2020).

Figura 4 – Envio de e-mail utilizando o Node Mailer com o Gmail.

```
const nodemailer = require('nodemailer');

const transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: 'servidor@servidor.com',
        pass: '[redacted]'
    }
});

const mailOptions = {
    from: 'remetente@servidor.com',
    to: 'destinatario@servidor.com',
    subject: 'Título do E-mail',
    text: 'Corpo do E-mail'
}

transporter.sendMail(mailOptions, (error, info) => {
    if (error) {
        console.log('Error:' + error);
    }
});
```

Fonte: O autor, 2020.

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

Integrando-se ao Node Mailer foi utilizado outro módulo do Node.Js, o Node - Cron. Esse pequeno módulo consegue agendar e executar tarefas agendadas (que são abstraídas em funções Javascript) usando a sintaxe do GNU Crontab (NPMJS, 2020). Podemos agendar uma determinada tarefa para uma determinada data e horário ou ciclicamente. Com o Crontab temos todas as facilidades para agendar tarefas repetidas e todas as facilidades necessárias ao agendamento de execução de scripts (NEVES, 2008). Com essas duas ferramentas é possível definir as funções que enviam os e-mails e então agendá-las para executarem periodicamente (FIGURA 5).

Figura 5 – Agendamento dos envios de e-mail com sintaxe GNU Crontab.

```
25
26  //Agendador de tarefas;
27  const cron = require('node-cron');
28
29  //Programa o envio de emails. O mesmo intervalo informado aqui deve ser informado na function;
30  cron.schedule('27 19 * * *', EmailsControllers.sendQuinquenniunsEmails);
31
32  cron.schedule('26 15 * * *', EmailsControllers.sendFruitionsBeginEmails);
33
34  cron.schedule('26 15 * * *', EmailsControllers.sendFruitionsEndEmails);
35
36  //Sintaxe do GNU Crontab que deve ser usada.
37  //* (segundos, 0-59, opcional).
38  //* * (minutos, 0-59, obrigatório).
39  //* * * (horas, 0-23, obrigatório).
40  //* * * * (dia do mês, 0-31, obrigatório).
41  //* * * * * (mês, 0-12, obrigatório).
42  //* * * * * * (dia da semana, 0-7, obrigatório).
43
44
```

Fonte: O autor, 2020.

2.4 Moment.Js

A biblioteca Moment.Js foi desenhada para funcionar no navegador (como uma importação comum Javascript) ou como um módulo do Node.Js Ela provê métodos de manipulação de datas como: *add*, *subtract*, *difference* e *inBetween* (MOMENT.JS, 2020).

Para esse projeto, em vez de utilizar a biblioteca Date, padrão do Javascript, que pode apresentar comportamentos diferentes de acordo com a plataforma em que a linguagem está sendo interpretada, foi utilizada a biblioteca Moment.Js, que encapsula a biblioteca padrão e a manipula de maneira eficiente e confiável. Outro aspecto positivo é a precisão com relação às operações matemáticas realizadas com as datas, que não se apoiam em valores fixos como 30

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

dias para um mês ou 365 dias para um ano, levando em consideração o calendário atual ao fazer os cálculos.

A Figura 6 mostra um exemplo prático de como a biblioteca foi utilizada no projeto para manipular datas e performar os cálculos necessários para identificar os quinquênios de determinado colaborador.

Figura 6 – Trecho da função usando Moment.Js para calcular os quinquênios.

```
async timeForNextQuinquenniumLocal(user_id) {
    const employeeFound = await connection('employee').select('*').where({ id: user_id }).first();
    if (employeeFound) {

        let employeeFoundAdmission = moment(employeeFound.admission, 'YYYY-MM-DD');

        let contribuitionTime = {
            inDays: moment().diff(employeeFoundAdmission, 'days', true),
            inMonths: moment().diff(employeeFoundAdmission, 'months', true),
            inYears: moment().diff(employeeFoundAdmission, 'years', true),
        }

        let quinquennium = 1;
        let quinqueniuns = [];

        let deltaDays = moment(employeeFoundAdmission).add(quinquennium * 5, 'years').diff(employeeFoundAdmission, 'days');
        quinqueniuns.push({
            quinquennium: quinquennium,
            beginDate: employeeFoundAdmission.format('YYYY-MM-DD'),
            endDate: moment(employeeFoundAdmission).add(quinquennium * 5, 'years').format('YYYY-MM-DD')
        });

        while (deltaDays < contribuitionTime.inDays) {
            quinquennium++;

            quinqueniuns.push({
                quinquennium: quinquennium,
                beginDate: moment(quinqueniuns[quinqueniuns.length - 1].endDate).format('YYYY-MM-DD'),
                endDate: moment(employeeFoundAdmission).add(quinquennium * 5, 'years').format('YYYY-MM-DD')
            });

            deltaDays = moment(employeeFoundAdmission).add(quinquennium * 5, 'years').diff(employeeFoundAdmission, 'days');
        };
```

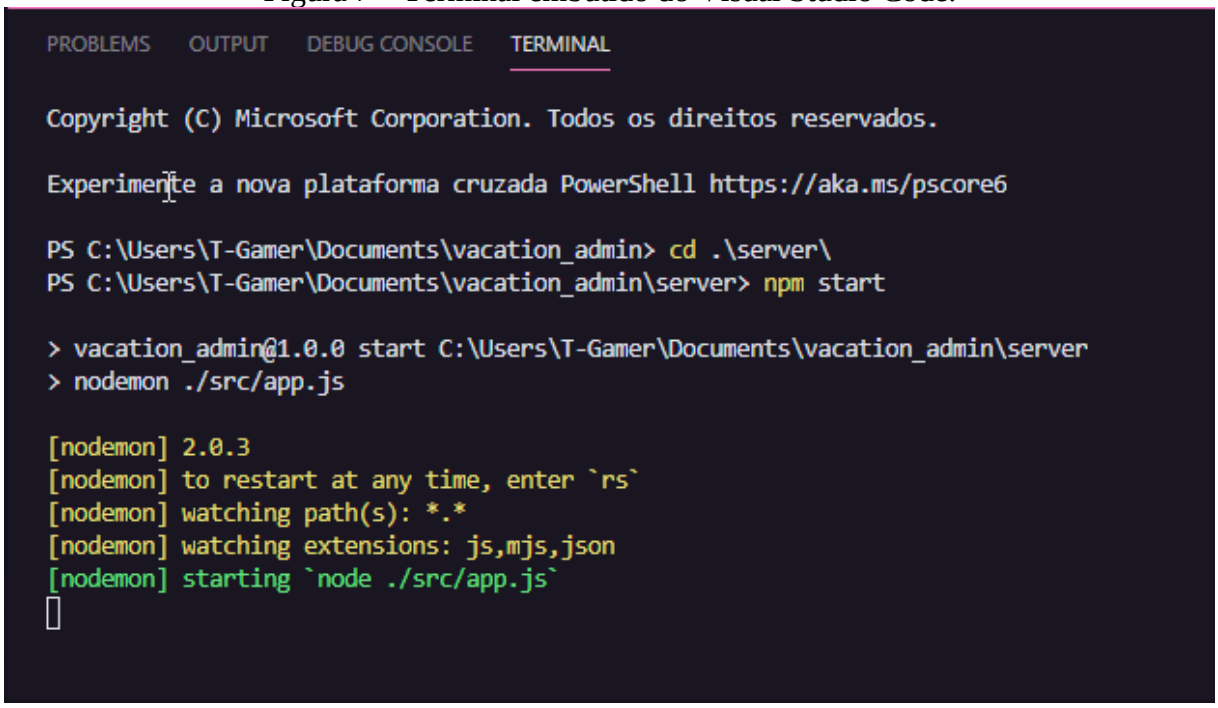
Fonte: O autor, 2020.

2.5 Visual Studio Code

O Visual Studio Code é um editor de código fonte leve, mas poderoso e que funciona em plataformas *desktop* e está disponível para Windows, macOS e Linux. Fornece suporte nativo para Javascript e Node.Js além de possuir um ecossistema repleto de extensões para outras linguagens (VISUALSTUDIO CODE, 2020). Nesse projeto o Visual Studio Code foi usado do começo ao fim. Apesar de não se tratar de um ambiente de desenvolvimento integrado, suas extensões e o seu terminal embutido (FIGURA 7) tornaram possível a compilação e a execução do servidor local onde o projeto foi desenvolvido, a visualização da estrutura do banco de dados e a execução consultas e visualização dos retornos do banco de dados para propósitos de *debug*.

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

Figura 7 – Terminal embutido do Visual Studio Code.

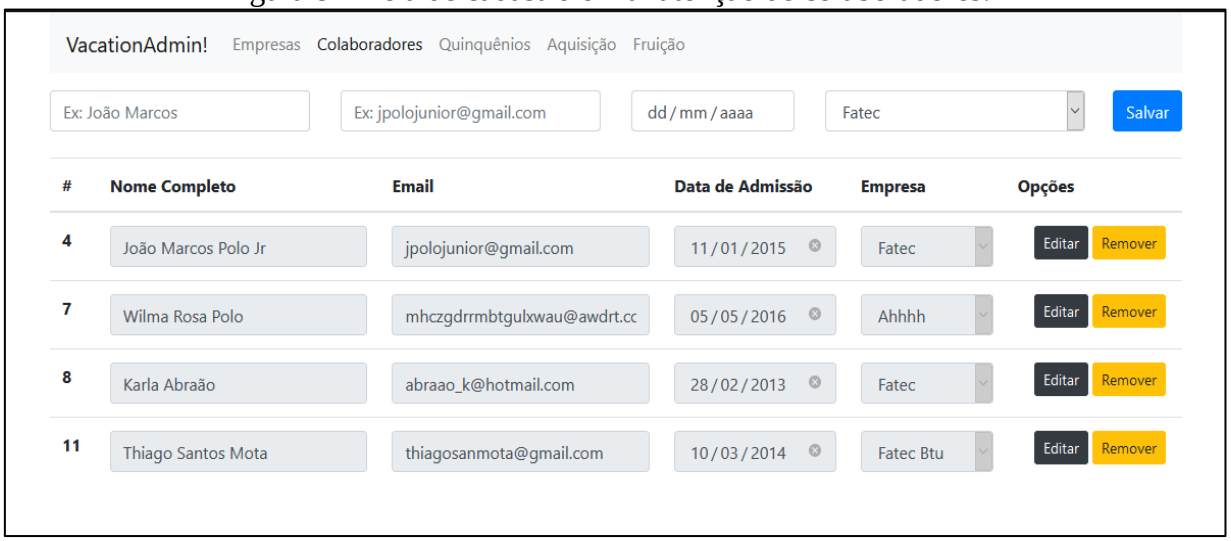


Fonte: O autor, 2020.

3 RESULTADOS E DISCUSSÃO

A aplicação *web* não possui uma página inicial predefina, porém, todas as páginas podem ser acessadas de qualquer lugar. Para que os usuários acostumados com planilhas possam se adaptar e a fim de reduzir a resistência ao uso da aplicação, as páginas (interfaces gráficas) foram desenvolvidas em um estilo de tabela (FIGURA 8) e todas elas são muito similares.

Figura 8 – Tela de cadastro e manutenção de colaboradores.



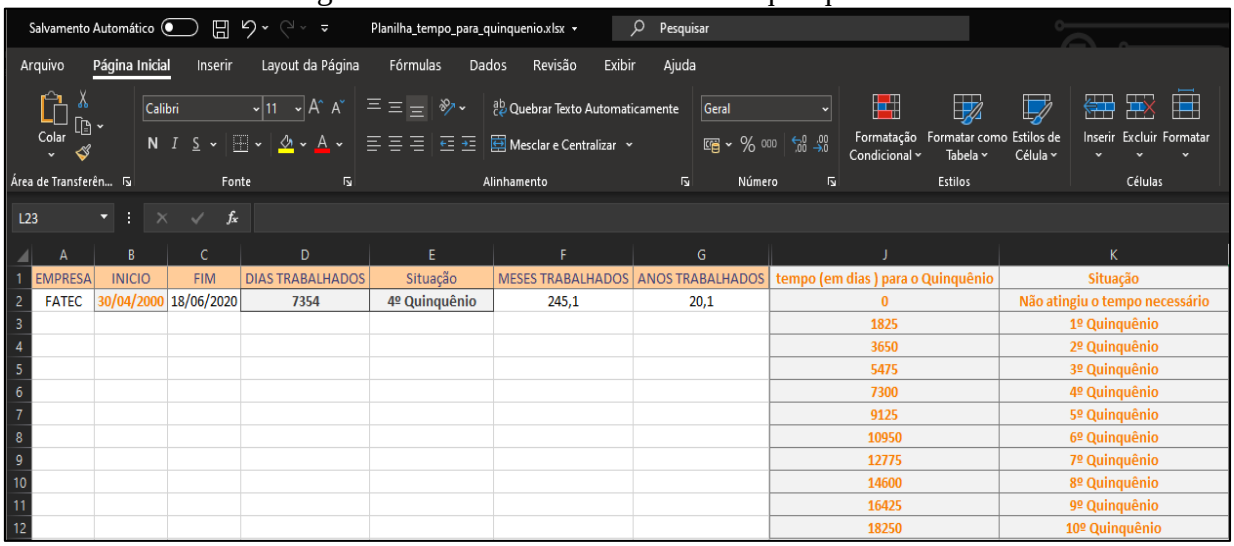
Fonte: O autor, 2020.

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

Ainda na Figura 8 pode-se observar a presença de um formulário que segue os mesmos tamanhos e campos da tabela, dando a ideia de que existe sempre uma célula aberta para inserção de registros.

As planilhas utilizadas eram responsáveis por realizar os cálculos dos quinquênios (FIGURA 9) e a manutenção dos períodos aquisitivos e fruitivos (FIGURA 10). Sendo assim, a aplicação também possuí além das páginas cadastrais, páginas onde o usuário pode obter os mesmos dados disponíveis nas planilhas, usufruindo da consistência de dados e validação deles (FIGURA 11).

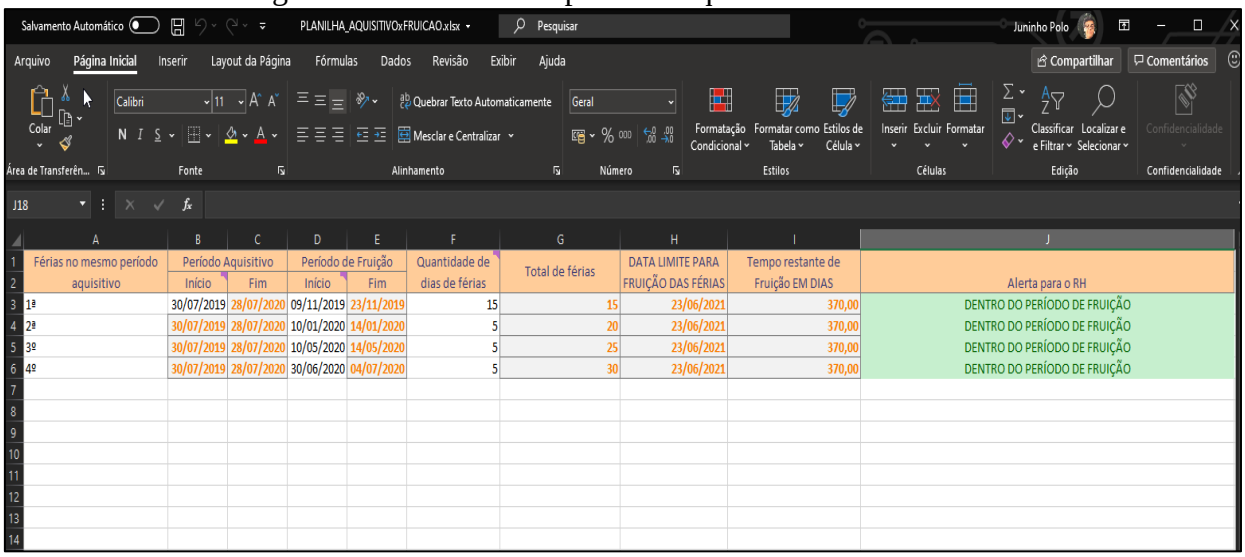
Figura 9 – Planilhas de cálculos dos quinquênios.



	A	B	C	D	E	F	G	J	K
1	EMPRESA	INICIO	FIM	DIAS TRABALHADOS	Situação	MESES TRABALHADOS	ANOS TRABALHADOS	tempo (em dias) para o Quinquênio	Situação
2	FATEC	30/04/2000	18/06/2020	7354	4º Quinquênio	245,1	20,1	0	Não atingiu o tempo necessário
3								1825	1º Quinquênio
4								3650	2º Quinquênio
5								5475	3º Quinquênio
6								7300	4º Quinquênio
7								9125	5º Quinquênio
8								10950	6º Quinquênio
9								12775	7º Quinquênio
10								14600	8º Quinquênio
11								16425	9º Quinquênio
12								18250	10º Quinquênio

Fonte: O autor, 2020.

Figura 10 – Planilhas de períodos aquisitivos e fruitivos.

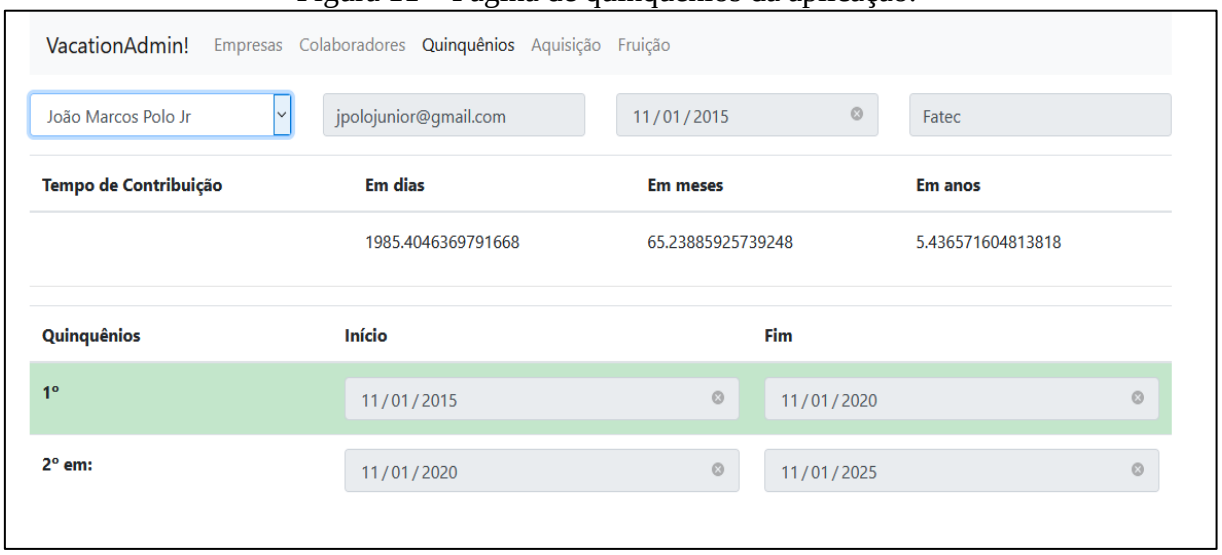


	A	B	C	D	E	F	G	H	I	J
1	Férias no mesmo período aquisitivo	Período Aquisitivo		Período de Fruição		Quantidade de dias de férias	Total de férias	DATA LIMITE PARA FRUIÇÃO DAS FÉRIAS	Tempo restante de Fruição EM DIAS	Alerta para o RH
2		Início	Fim	Início	Fim					
3	1ª	30/07/2019	28/07/2020	09/11/2019	23/11/2019	15	15	23/06/2021	370,00	DENTRO DO PERÍODO DE FRUIÇÃO
4	2ª	30/07/2019	28/07/2020	10/01/2020	14/01/2020	5	20	23/06/2021	370,00	DENTRO DO PERÍODO DE FRUIÇÃO
5	3º	30/07/2019	28/07/2020	10/05/2020	14/05/2020	5	25	23/06/2021	370,00	DENTRO DO PERÍODO DE FRUIÇÃO
6	4º	30/07/2019	28/07/2020	30/06/2020	04/07/2020	5	30	23/06/2021	370,00	DENTRO DO PERÍODO DE FRUIÇÃO

Fonte: O autor, 2020.

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

Figura 11 – Página de quinquênios da aplicação.



Fonte: O autor, 2020.

Foram definidos três envios de e-mails. A biblioteca de e-mails em conjunto com o agendador de tarefas faz checagens periódicas para enviar os e-mails quando as condições codificadas são satisfeitas. Para os quinquênios, no começo do mês, uma tarefa checa se algum quinquênio se completará no mês seguinte, então obtém os dados do usuário e envia uma notificação por e-mail (FIGURA 12).

Figura 12 – Envio de e-mail de quinquênio

Fonte: O autor, 2020.

Para os períodos fruitivos, diariamente faz-se uma checagem por períodos fruitivos que se iniciarão ou terminarão, enviando os e-mails para os seus respectivos colaboradores

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

(FIGURA 13). Essa sincronia entre a frequência com que as tarefas executam as condições que os dados precisam satisfazer garante que nenhum e-mail deixe de ser enviado.

Figura 13 – Alerta de término de período fruitivo.

Fonte: O autor, 2020.

Toda essa integração do Javascript no desenvolvimento do *front-end* e do *back-end* promoveram uma maior produtividade, uma vez que não é preciso ficar trocando de linguagem toda vez que é necessário desenvolver uma página ou um procedimento. Entretanto, conforme o código do *front-end* cresce em tamanho ou complexidade devido às diversas funcionalidades de uma página dinâmica, o Javascript, sem a adição de um *framework* de desenvolvimento *front-end* deixa de ser a melhor opção.

Para esse desenvolvimento, as páginas *web* foram codificadas de forma básica, somente com seu esqueleto. Ao serem entregues do servidor ao cliente, o Javascript tem a tarefa de requisitar e então popular os dados persistidos no banco e manipulados pelo servidor. Todavia, realizar essas tarefas usando somente Javascript, ou até mesmo Jquery se torna um processo custoso e trabalhoso.

4 CONCLUSÕES

Muito do que torna um sistema único são suas regras de negócio. Cabe ao analista ou desenvolvedor interagir com o usuário e entender essas regras, promovendo não somente a eficácia do sistema, mas também a eficiência dos processos. O sistema desenvolvido garantiu esses pontos, transformando planilhas que apresentavam uma série de funções do Excel (procv, se, etc.) em um sistema enxuto que facilita a gestão dos quinquênios, períodos fruitivos e aquisitivos e, ainda, realiza o envio automático de e-mails para os usuários configurados quando os prazos são atingidos. Com isso, o sistema desenvolvido diminui a chance de o usuário cometer erros devido a série de informações na planilha que são interdependentes. Algumas tarefas que antes eram feitas manualmente, foram automatizadas e é possível garantir que uma

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

regra de negócio vai continuar intacta, pois está codificada no sistema e não apenas escrita em uma célula.

Com certeza será necessário um acompanhamento do uso do sistema pelos usuários. Isso apontará erros, melhorias ou necessidades que podem ser codificadas em uma próxima versão. A utilização de um framework para desenvolvimento *front-end* baseado em componentes ou o uso de uma tecnologia mais tradicional para transferência de conteúdo entre servidor e cliente como react.js ou cakephp, respectivamente, é algo que definitivamente pode ser considerado em uma próxima versão.

REFERÊNCIAS

BOTUCATU, **Estatuto do Servidor Público do Município** (2011), Seção IX – Do Licença Prêmio, Art. 114. Disponível em: https://leismunicipais.com.br/estatuto-do-servidor-funcionario-publico-botucatu-sp. Acesso em: 18 de nov. 2020.

BRASIL. **Consolidação das Leis do Trabalho.** Decreto-Lei nº 5.452, de 01 maio 1943. Disponível em: https://www.jusbrasil.com.br/topicos/10754675/artigo-130-do-decreto-lei-n-5452-de-01-de-maio-de-1943. Acesso em: 18 jun. 2020.

CARVALHO, José Oscar Fontanini de. O papel da interação humano-computador na inclusão digital. **Transinformação**, v. 15, n. SPE, p. 75-89, 2003. Disponível em: https://www.scielo.br/pdf/tinf/v15nspe/04.pdf. Acesso em: 18 jun. 2020.

FELIPINI, Dailton. **Email marketing eficaz**: Como conquistar e fidelizar clientes com uma newsletter. São Paulo: LeBooks Editora, 2014. p. 4-15. Disponível em: https://books.google.com.br/books?id=FpP9AwAAQBAJ&lpg=PT2&dq=marketing%20com%20email&lr=lang_pt&hl=pt-BR&pg=PT1#v=onepage&q=marketing%20com%20email&f=false. Acesso em: 10 mar. 2020.

FLANAGAN, David. **Javascript**: O guia definitivo. Santana: O'REILLY, 2013. Cap. 1. p. 1-8. Disponível em: https://books.google.com.br/books?id=zWNyDgAAQBAJ&lpg=PR1&ots=IzDdB1PckQ&dq=javascript&lr&hl=pt-BR&pg=PR1#v=onepage&q=javascript&f=false. Acesso em: 17 jun. 2020.

HAHN, Evan. **Express in Action:** Writing, building, and testing Node. js applications. Nova Iorque: Manning Publications, 2016. Cap. 1. p. 3-17. Disponível em: https://www.hackerstribe.com/wp-content/uploads/2016/04/Node.js-Express-in-Action.pdf. Acesso em: 17 jun. 2020.

KNEX.JS. **KNEX.JS**. 17 jun. 2020. Disponível em: http://knexjs.org/#changelog. Acesso em: 17 jun. 2020.

KREIBICH, Jay. **Using SQLite**. Sebastopol: O'REILLY, 2010. Cap. 1. p. 1-6. Disponível em:

Tekhne e Logos, Botucatu, SP, v.13, n.2, setembro, 2022.
ISSN 2176 – 4808

https://books.google.com.br/books?id=HFIM47wp0X0C&lpg=PR7&ots=Fh5wiVqm5S&dq=sqlite&lr&hl=pt-BR&pg=PR7#v=onepage&q=sqlite&f=false. Acesso em: 17 jun. 2020.

MASSE, Mark. **REST API Design Rulebook:** Designing Consistent RESTful Web Service Interfaces. Sebastopol: O'REILLY, 2011. Cap. 1. p. 1-7. Disponível em: https://books.google.com.br/books?id=eABpzyTcJNIC&lpg=PR3&ots=vzXA_-i8GB&dq=api%20rest&lr&hl=pt-BR&pg=PR3#v=onepage&q=api%20rest&f=false. Acesso em: 17 jun. 2020.

MOMENT.JS. **Where to use it**. 15 mar. 2020. Disponível em: https://momentjs.com/docs/. Acesso em: 15 jun. 2020.

NEVES, Julio Cezar. **Programação Shell Linux**. 7. ed. Rio de Janeiro: Brasport, 2008. Cap 9. p. 70-74. Disponível em: https://books.google.com.br/books?id=BYS26h1v9OYC&lpg=PA3&ots=Teqd6e0kKb&dq=linux%20cron&lr&hl=pt-BR&pg=PR25#v=onepage&q=cron&f=false. Acesso em: 18 jun. 2020.

NODE.JS. **About Node.js**. 17 jun. 2020. Disponível em: https://nodejs.org/en/about/. Acesso em: 17 jun. 2020.

NODEMAILER. **NODEMAILER**. 01 jun. 2020. Disponível em: https://nodemailer.com/about/. Acesso em: 01 jun. 2020.

NPMJS. **NODE-CRON:** Getting Started. 01 jun. 2020. Disponível em: https://www.npmjs.com/package/node-cron. Acesso em: 01 jun. 2020.

PEREIRA, Caio Ribeiro. **Aplicações web real-time com Node.js**. São Paulo: Casa do Código, 2014. Cap. 1. p. 9-23. Disponível em: https://books.google.com.br/books?id=Wm-CCwAAQBAJ&lpg=PT3&ots=_duZ1mzseO&dq=node%20js&lr&hl=pt-BR&pg=PT3#v=onepage&q=node%20js&f=false. Acesso em: 17 jun. 2020.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software**. São Paulo: McGraw Hill Brasil, 2016. Cap. 1. p. 1-13. Disponível em: https://books.google.com.br/books?id=wexzCwAAQBAJ&lpg=PR1&ots=0N2GoMJt_X&dq=engenharia%20de%20software&lr&hl=pt-BR&pg=PR11#v=onepage&q=engenharia%20de%20software&f=false. Acesso em: 18 jun. 2020.

SANTOS, Clayton Ferreira. **Planilha eletrônica (Excel)**. Universidade Federal de Lavras. Minas Gerais, 2011. Disponível em: https://docplayer.com.br/3318341-Planilha-eletronica-excel-instrutor-clayton-ferreira-santos.html. Acesso em 18 jun. 2020.

UZAYR, Sufyan Bin; CLOUD, Nicholas; AMBLER, Tim. **JavaScript Frameworks for Modern Web Development**: the essential frameworks, libraries, and tools to learn right now. 2. ed. Nova Iorque: Apress, 2019. Cap 10. p. 377-426.

VISUALSTUDIO CODE. **Getting Started**. 01 jun. 2020. Disponível em: https://code.visualstudio.com/docs. Acesso em: 01 jun. 2020.